VXML Server Configuration

VXML Server can be configured to modify its function. This chapter explains all configuration options and how to change them.

VXML Server uses default values for these configuration options and functions without modification. Improperly chosen values can cause significant performance degradations and could even prevent VXML Server from functioning correctly.

Note Only an experienced administrator should consider changing these options.

Global Configuration File

The method to edit the VXML Server configuration is through an XML file named `global_config.xml` found in the `%CVP_HOME%\VXMLServer\conf` directory. This file must be edited by hand; there is no graphical interface.

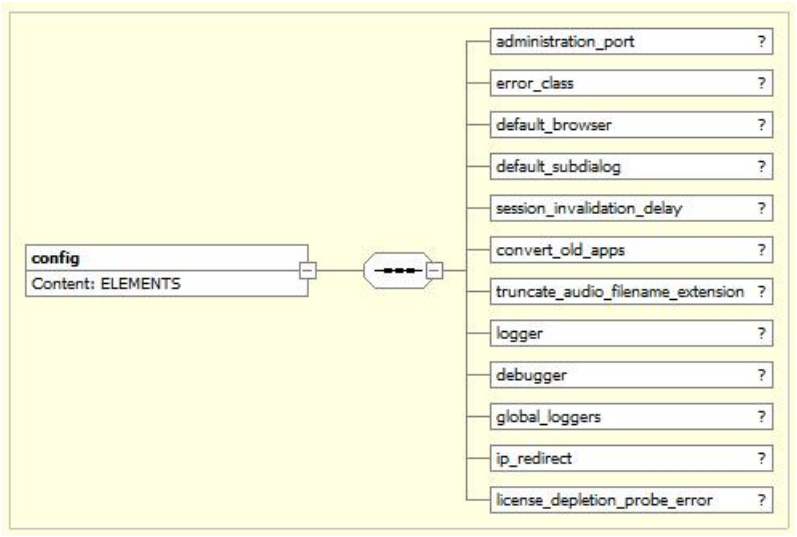
This file is loaded by VXML Server when it is initialized and cached in memory. Loading the file is one of the first tasks performed by VXML Server when it starts up because the configuration options affect how it runs. Any changes to this file requires VXML Server to be restarted in order for the changes to take affect.

Note When performing an upgrade of VXML Server, the administrator will have to reimplement the configuration changes.

Setup Options

The following figure displays the DTD diagram of the `global_config.xml` file.

Figure 1: Global Configuration Options



The elements in the XML document are:

- **administration_port**—This tag defines the port on which administration activity takes place and can be any positive integer. By default, the port is set to 10100. See Administration for more on administration activities.
- **error_class**—This tag defines the fully qualified Java class name of a class to run when an error occurs for notification purposes. By default no class is defined. See Programming Guide for Cisco Unified CVP VXML Server and Unified Call Studio for more on how to write the On Error Notification class.
- **default_browser**—This tag defines the real name of the gateway adapter that should be used by default when VXML Server needs to produce a VoiceXML page in a scenario where the current application is unknown and therefore the gateway adapter for that application is unknown. One such scenario is an error where the VXML Server session is unrecognized. The reason this exists is because some gateways require the VoiceXML to be formatted in a specific way (such as requiring an XML namespace to appear in the document) that if the VoiceXML page were produced in a different format would cause an error on the gateway. An application lists its gateway in its settings and generally this is available to VXML Server to produce the correct VoiceXML. However, in rare cases, an error occurs and VXML Server does not have access to the session and the application that the call belonged to needs to know which gateway to have the resulting VoiceXML page conform to. By default, if left blank in `global_config.xml`, VXML Server will search through the directory of installed gateway adapters and use the first one it finds.
- `default_subdialog`—This tag defines whether to treat a call that is not associated with an application as if it were a VoiceXML subdialog and whose possible values are `true` and `false`. Some gateways (such as Cisco gateways) call all VXML Server applications as VoiceXML subdialogs. VXML Server must be aware of this because it determines how the VoiceXML it produces looks and if not produced correctly would cause an error on the gateway. Typically, a call is made to an application which defines in its settings whether to treat the application as a subdialog. However, in rare cases, an error occurs and VXML Server does not have access to the session and the application that the call belonged to needs to know whether to treat the call as a subdialog. By default, if left blank in `global_config.xml`, VXML Server will consider a call to the application to **not** be a subdialog.

- `session_invalidation_delay`—This tag defines the amount of time in seconds that VXML Server will wait for after a call session ends before actually invalidating that session (this can be any integer greater than or equal to 0). This configuration option is necessary because there may be various activities taken by loggers and end of call classes that require the session to remain alive to access data within it (such as element or session data), and if the session was invalidated errors can occur when attempting to access the data. If this value were too small (such as 0 seconds), many errors can occur for routine actions such as logging at the end of a call. If this value was too high, too many sessions would remain in memory for too long, potentially causing memory issues. We highly recommend keeping the default value of 30 seconds, or testing the system if this value changes.
- `convert_old_apps`—This tag defines whether to convert applications deployed from a version of Call Studio that VXML Server detects is old (possible values are `true` and `false`). By setting this configuration option to true, a deployed application can be copied to the `applications` directory of VXML Server without requiring the application to be redeployed from the latest version of Call Studio.

Note For new application settings, the converter will choose default values.

Note This converter is limited to converting the XML files that define an application with regards to Call Studio and VXML Server and will **not** convert any other files or Java classes for the application. By default this configuration option is on.

- **logger**—This tag acts as the parent tag for three additional tags that relate to loggers. The first two tags, `<minimum_thread_pool_size>`, and `<maximum_thread_pool_size>` define the minimum and maximum size of the thread pool that is used for handling logger threads. The minimum thread pool size value can be any positive integer and the maximum thread pool size value can be any positive number as long as it is greater than the minimum thread pool size value.

Note If the maximum number of logger threads is used, VXML Server will queue the logger events to be used when a thread becomes available so the data will not be lost. Because these values affect thread usage, we highly recommend that any deviation from the default values (1 minimum / 500 maximum) be fully tested for any complications. For example, if the maximum is set to a *low* value and the system encounters high load, VXML Server might encounter a situation where the queued logger events accumulate faster than the logger threads can handle them, leading to a scenario where the application server runs out of memory. If the maximum value is set *too high* and the system encounters high load, the system on which VXML Server runs might run out of threads to allocate, which can cause many other problems with the application server as well as the operating system. Of all the VXML Server configuration options, these two have the highest potential for causing major problems if misused.

The third child tag, `<keep_alive_time>` defines the amount of time in seconds that a thread should be idle before it is removed from the thread pool. This tag allows for the thread pool size to decrease over time as logger volume decreases. This value allows for optimum thread pool size based on the call volume.

The default value is 30 seconds. We recommend not change drastically from the default becasue too high a number will keep unnecessary resources around, and too low a number will reduce efficiency and defeat the purpose of using a thread pool completely. Refer to VXML Server Logging for more on logging.

- **debugger**—This tag defines the RMI registry port for the Call Studio debugger. This configuration option is used only by VXML Server implementations used by Call Studio for debugging purposes and should not be used in a production environment. The default is 8099 and the value can be any positive integer.
- **global_loggers**—This tag defines the global loggers to use within VXML Server. Administrators can add additional global loggers as well as change or remove the loggers listed by default: the global call, admin, and error loggers. Each logger instance is defined by a separate child tag `<logger_instance>`. The required `name` attribute gives the logger instance a name and must be unique. The required `class` attribute gives the fully qualified Java class name that defines the global logger. The optional `configuration` attribute gives the name of a configuration file for the global logger if needed. This configuration file is expected to reside in the same `%CVP_HOME%\VXMLServer\conf` directory. Refer to VXML Server Logging for more on logging and the *Programming Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio*for more on creating custom loggers.
- **ip_redirect**—This tag enables a feature that redirects the gateway to point directly to the VXML server, which circumvents any load balancers that may lie between the gateway and server. This tag uses a VoiceXML goto with the explicit IP address of the server for the first VoiceXML page of a call. When this feature is enabled, the VXML server uses a load balancer only for the first HTTP request of a call session, but not for subsequent requests for the same call session. The tag, if used, takes a required attribute `active` which can be *true* or *false*. It also takes an optional attribute, `ip_to_use`, which contains the IP address to use for a redirect if you do not want to use the IP address returned by the default VXML server.

  This feature minimizes the effect of load-balancer related issues by using the load balancer only for the first HTTP request of a call session. For example, if a loadbalancer fails, then a VXML Server may incur many session timeouts. Because a load balancer failure affects all VXML servers that it handles, a downward effect can quickly occur where sessions are stale and calls are placed on hold across all VXML servers in the pool.

Note If this feature is enabled, you do not need to configure sticky cookies on the load balancers.

- **license_depletion_probe_error**—This tag defines the response to a probe when licenses have been depleted. If this configuration is used, and the body of the tag is set to *true*, then an HTTP 500 error is returned when licenses are depleted. This tag ensures that loadbalancers do not continue to send traffic to the server. If this tag is not used, or the body of the tag is set to *false*, then a HTTP 200 response is sent back to probes. This feature must only be used if the `ip_redirect` feature is set to *active*.